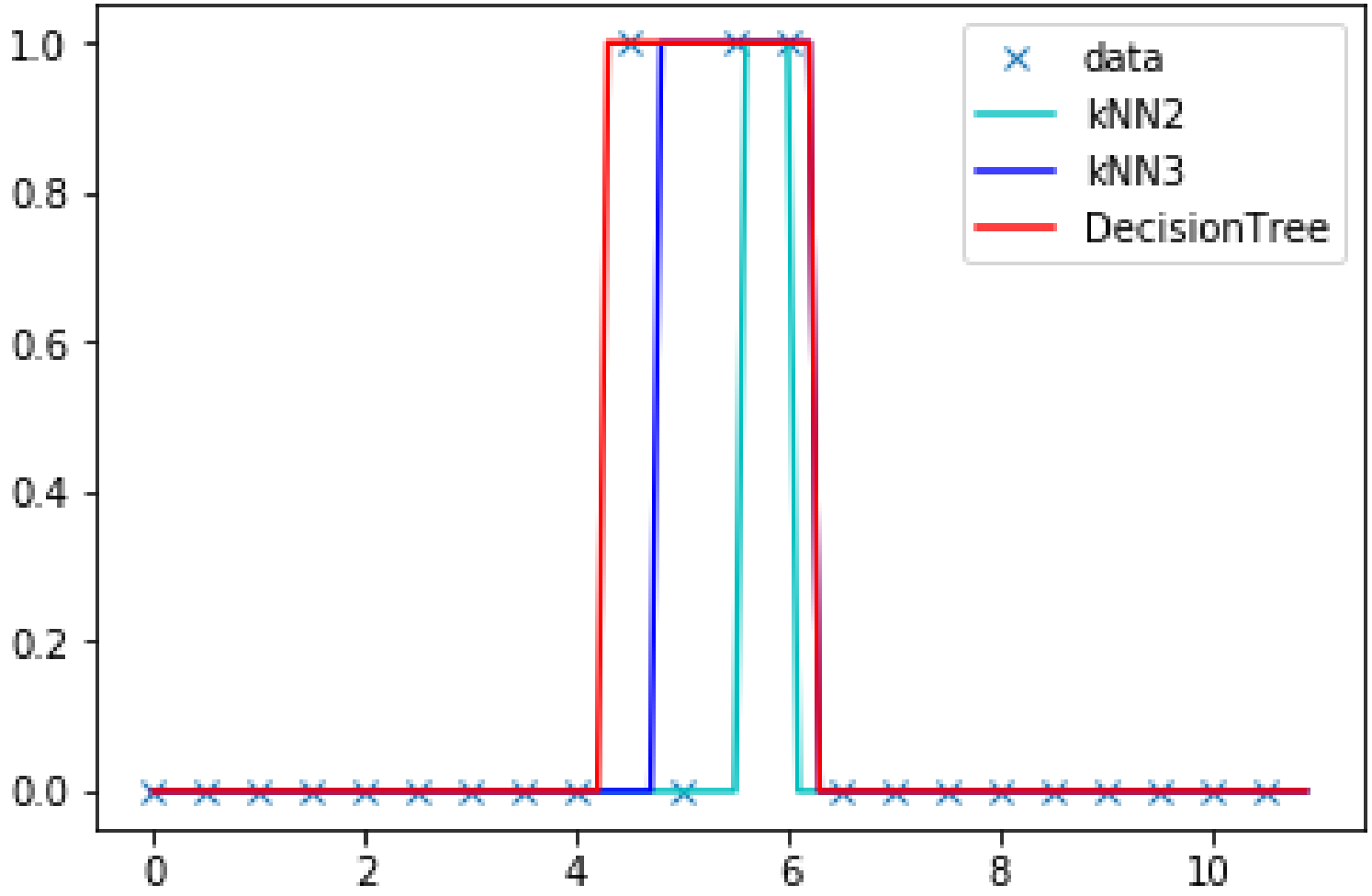
Tutorial

https://docs.scipy.org/doc/numpy-dev/user/quickstart.html

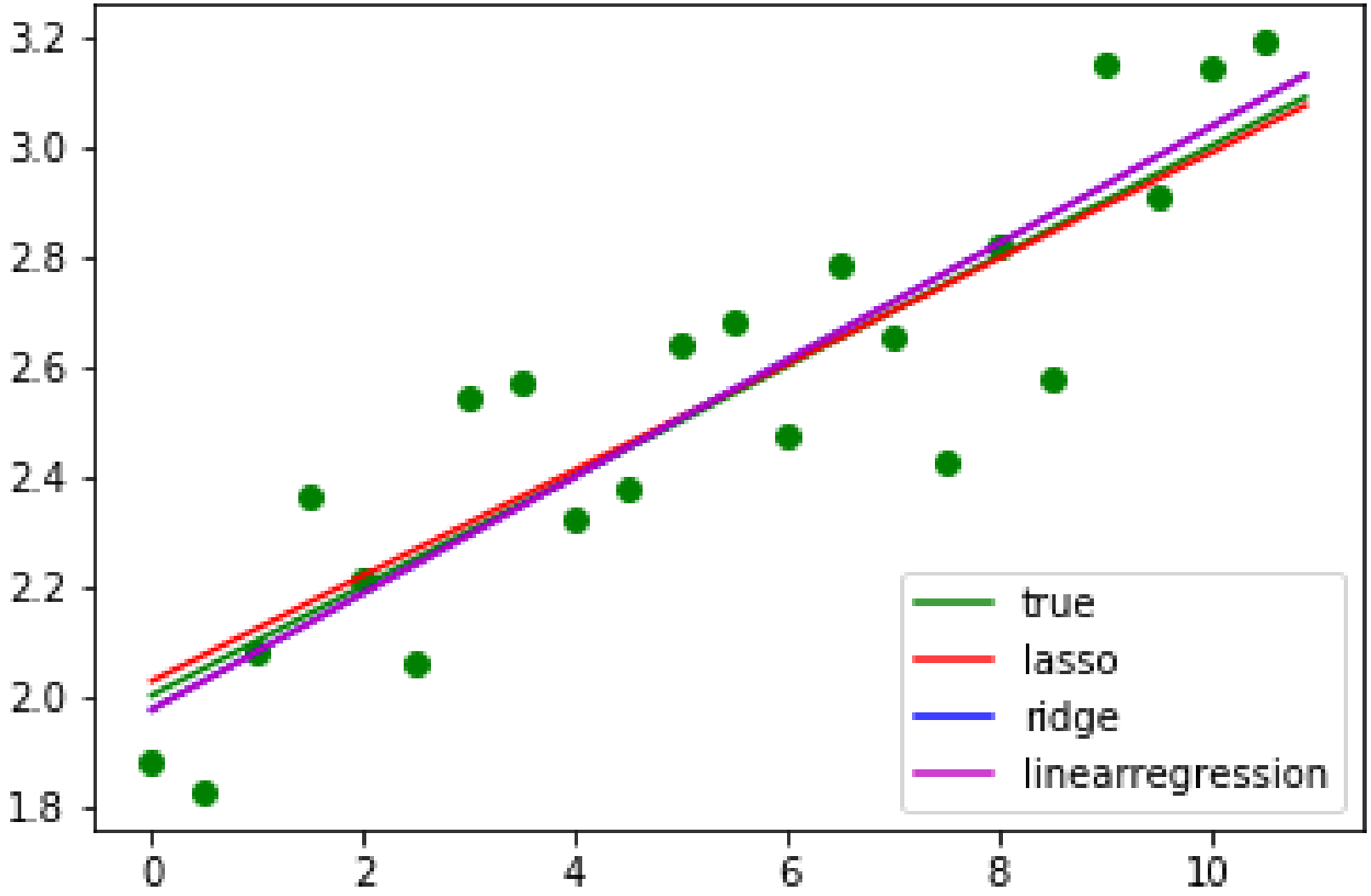
http://scikit-learn.org/stable/

Intro

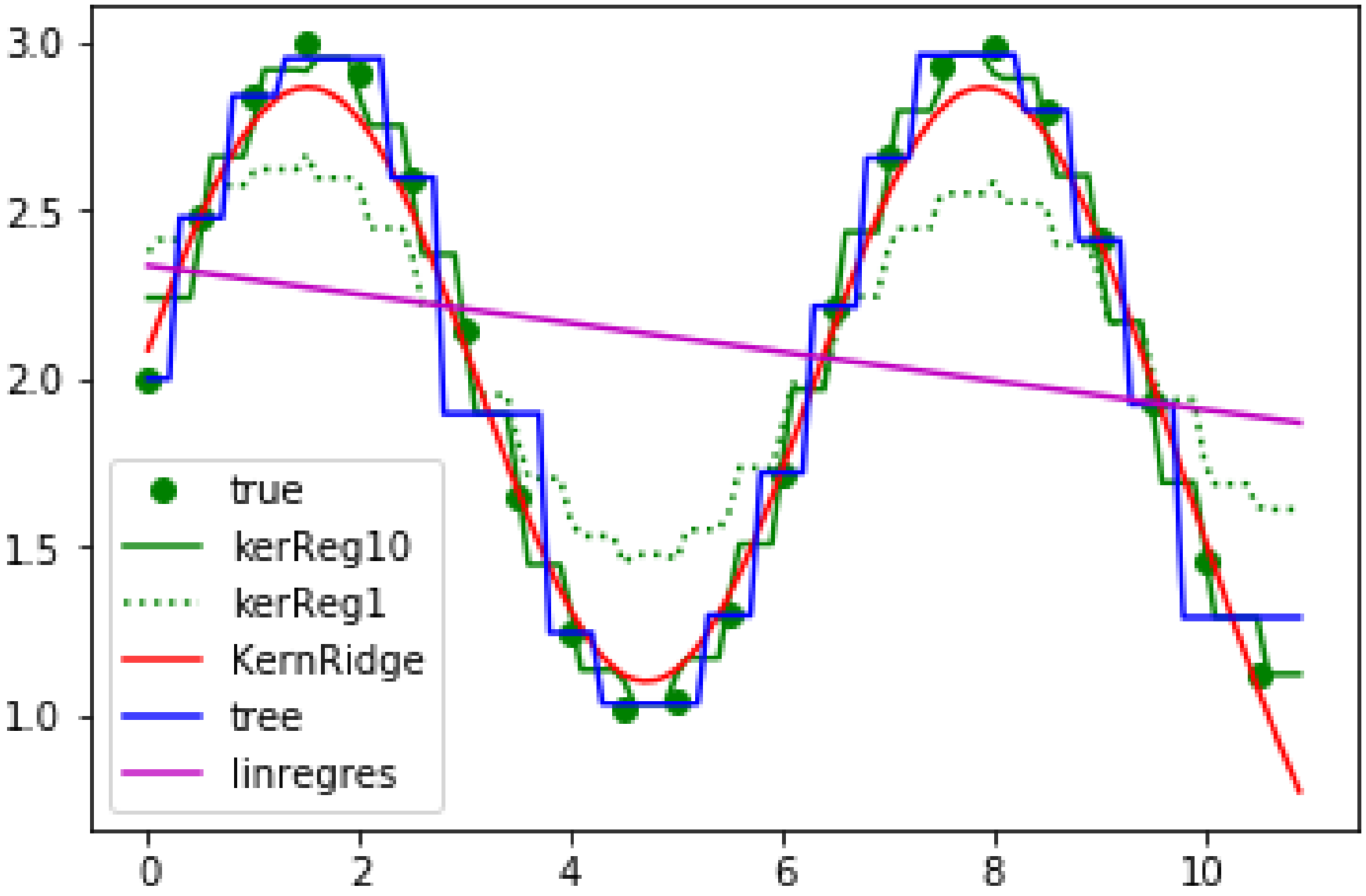
```
from sklearn import neighbors, datasets
import numpy as np
from sklearn import tree
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
#Training Set
X = np.zeros((22,1))
X[:,0] = np.arange(0,11,.5)
noisesigma = 2.5
Y = np.ravel((2-(X-5)**2 + noisesigma * np.random.randn(22, 1))>0)
#Testing Set
Xp = np.zeros((110,1))
Xp[:,0] = np.arange(0,11,.1)
#KNeighborsClassifier
n_neighbors = 2
weights = 'distance'
weights = 'uniform'
clfKNNC = neighbors.KNeighborsClassifier(n_neighbors, weights=weights)
clfKNNC.fit(X, Y)
#KNeighborsClassifier
n_neighbors = 3
weights = 'distance'
weights = 'uniform'
clfKNNC2 = neighbors.KNeighborsClassifier(n_neighbors, weights=weights)
clfKNNC2.fit(X, Y)

min_samples_split = 8
clftree = tree.DecisionTreeClassifier(min_samples_split=min_samples_split)
clftree = clftree.fit(X, Y)
YpKNNC = clfKNNC.predict(Xp)
YpKNNC2 = clfKNNC2.predict(Xp)
Yptree = clftree.predict(Xp)
#clf.predict_proba(Xp)
plt.plot(X,Y,'x',label='data')
plt.plot(Xp,YpKNNC,'c',label='kNN2')
plt.plot(Xp,YpKNNC2,'b',label='kNN3')
plt.plot(Xp,Yptree,'r',label='DecisionTree')
plt.legend( loc = 1 )
```


1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
data
kNN2
kNN3
DecisionTree

```
from sklearn import neighbors, datasets
import numpy as np
from sklearn import tree
from sklearn import linear_model
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap

#Training Set
X = np.zeros((22,1))
X[:,0] = np.arange(0,11,.5)
noisesigma = .2
Y = np.ravel(2 + .1 * X + noisesigma * np.random.randn(22, 1))

#Testing Set
Xp = np.zeros((110,1))
Xp[:,0] = np.arange(0,11,.1)
Yp = np.ravel(2 + .1 * Xp)
# Linear Regression
reglr = linear_model.LinearRegression()
reglr.fit(X,Y)
Ylr = reglr.predict(Xp)
regridge = linear_model.RidgeCV(alphas=[0.1])
regridge.fit(X,Y)
Yridge = regridge.predict(Xp)
reglasso = linear_model.Lasso(alpha = 0.1)
reglasso.fit(X,Y)
Ylasso = reglasso.predict(Xp)
plt.plot(X,Y,'go')
plt.plot(Xp,Yp,'g',label='true')
plt.plot(Xp,Ylasso,'r',label='lasso')
plt.plot(Xp,Yridge,'b',label='ridge')
plt.plot(Xp,Ylr,'m',label='linearregression')
plt.legend( loc = 4 )
```


3.2
3.0
2.8
2.6
2.4
2.2
2.0
1.8
0
2
4
6
8
10
true
lasso
ridge
linearregression

```
from sklearn import neighbors, datasets
import numpy as np
from sklearn import tree
from sklearn import linear_model
from sklearn.kernel_ridge import KernelRidge
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
def kernelregress(D,xq,beta):
    kk = np.exp(-beta*np.round(np.abs(D[:,0]-xq)))
    y = np.dot(kk,D[:,1])/np.sum(kk,1)
        return y

#Training Set
X = np.zeros((22,1))
X[:,0] = np.arange(0,11,.5)
noisesigma = 0
Y = (2 + np.sin(X) + noisesigma * np.random.randn(22, 1))
#Y[[5,10,15],0] = 2 * Y[[5,10,15],0]
#Testing Set
Xp = np.zeros((110,1))
Xp[:,0] = np.arange(0,11,.1)
Yp = (2 + np.sin(Xp))
# Linear Regression
reglr = linear_model.LinearRegression()
reglr.fit(X,Y)
Ylr = reglr.predict(Xp)
# Kernel Ridge Regression
regkr = KernelRidge(kernel='rbf', gamma=0.1,alpha=0.1)
regkr.fit(X,Y)
Ykr = regkr.predict(Xp)
# Kernel Regression
Yp1 = kernelregress(np.hstack((X,Y)),Xp,10)
Yp2 = kernelregress(np.hstack((X,Y)),Xp,1)
# Decision Tree Regressor
min_samples_split = 3
regtree = tree.DecisionTreeRegressor(min_samples_split=min_samples_split)
regtree = regtree.fit(X, Y)
Ytree = regtree.predict(Xp)
plt.plot(X,Y,'go',label='true')
plt.plot(Xp,Yp1,'g',label='kerReg10')
plt.plot(Xp,Yp2,'g:',label='kerReg1')
plt.plot(Xp,Ykr,'r',label='KernRidge')
plt.plot(Xp,Ytree,'b',label='tree')
plt.plot(Xp,Ylr,'m',label='linregres')
plt.legend( loc = 3 )
```


3.0
2.5
2.0
1.5
1.0
0
2
4
6
8
10
true
kerReg10
kerReg1
KernRidge
tree
linregres